Testes e Validação de Software 2017/18
Instituto Superior Técnico

Enunciado do Projecto

Data de Entrega: 6 de Abril de 2018

1 Introdução

O desenvolvimento de grandes sistemas de informação é um processo complexo e requer diversos níveis de abstracção. Um primeiro passo é a especificação do problema de uma forma rigorosa. Após a especificação rigorosa do problema, é possível efectuar provas de correcção do mesmo ou, caso este esteja incorrecto, descobrir falhas que de outra forma seriam difíceis de detectar. Depois, dever-se-á começar a codificar a solução e a testá-la simultaneamente por forma a detectar os erros o mais cedo possível.

O projecto de Testes e Validação de Software consiste no desenho de casos de teste a partir de uma especificação disponibilizada. Esta especificação diz respeito a algumas entidades do sistema informático de um operador de telecomunicações móveis (designado como *Vos*).

Este projecto tem como objectivo principal de aprendizagem que os alunos ganhem experiência no desenho de casos de testes ao aplicarem os padrões de teste leccionados nas aulas teóricas a um conjunto de classes e métodos cujo comportamento está descrito neste documento.

A secção 2 descreve as entidades que participam no sistema a testar assim como alguns detalhes de concretização destas entidades e que serão importantes para os casos de teste a desenhar. Finalmente, a secção 3 identifica os testes a realizar e a forma como a resolução do projecto vai ser avaliada.

2 Descrição do sistema

O sistema a testar é responsável pela gestão dos clientes e comunicações do operador de comunicações móveis *Vos*. Este sistema gere os clientes, números de telefones, telemóveis e telecomunicações que envolvem os clientes de *Vos*. Neste sistema, um cliente tem um número limitado de números de telefones. Cada número de telefone pode estar associado a um dado telemóvel. O sistema tem que manter a informação sobre as comunicações feitas por cada telemóvel por forma a ser capaz de determinar o custo das comunicações para cada cliente.

2.1 A entidade ClientManager

A entidade `ClientManager` é responsável por gerir os clientes de *Vos*. Esta entidade mantém os clientes registados e sabe todos os números de telefone (os já atribuídos a clientes e os que ainda estão livres). Cada cliente de *Vos* é identificado univocamente pelo seu número de identificação fiscal (um inteiro positivo com 9 dígitos). Cada número de telefone de *Vos* só pode estar atribuído no máximo a um cliente de *Vos*. A figura 1 apresenta a interface desta entidade. Neste contexto, pode considerar que um número de telefone válido é um número inteiro com 9 dígitos.

A atribuição de um número de telefone livre de *Vos* a um cliente de *Vos* é feita através da invocação do método `assignPhoneNumber`. A atribuição do número de telefone ao cliente em causa só é feita se todas as condições a verificar forem verdadeiras. Caso contrário, esta método apenas lançará a excepção `InvalidOperationException`.

O método `computeBill` é responsável por determinar o valor a pagar por um dado cliente de *Vos* tendo em conta as comunicações efectuadas pelo cliente através dos seus telemóveis registados. O cálculo a realizar é o seguinte:

```
public class  ClientManager {

  // Creates a new client of Vos and stores it. If phoneNumber is a number
  // already assigned or it is not valid, then InvalidOperationException is
  // thrown. If there is already a registered client with an identification
  // number equal to nif, then the DuplicateNIFException is thrown.
  public Client register(String clientName, int nif, int phoneNumber)
       throws DuplicateNIFException, InvalidOperationException { /* .... */ }

  // returns the client with the desired nif, null if not found
  public Client getClient(int nif) { /* .... */ }

  // Assigns a free phone number of Vos to the client of Vos identified by nif. If it is not
  // possible to assign the phone number, then the InvalidOperationException is thrown.
  public final void assignPhoneNumber(int clientNif, int phoneNumber) throws InvalidOperationException
   { /* .... */ }

  // computes the bill of the client taking into account his communications.
  public float computeBill(Client client) { /* .... */ }

  // returns the list of clients of Vos
  public List<Client> getClients()  { /* .... */ }

  ...
}
```

Figura 1: A interface da classe *ClientManager*.

- Se o número de chamadas efectuadas for inferior a 10, então o custo de cada chamada é $0,15$. O custo de cada SMS depende ainda do número de SMSs enviados. Se o cliente tiver enviado menos do que 300, o custo unitário é $0,05$, caso contrário é $0,045$.
- Se o número de chamadas realizadas for maior ou igual a 10 e inferior a 50, então o custo de cada chamada é $0,10$ e o custo de cada SMS depende ainda do número de SMSs enviados. Se o cliente tiver enviado menos do que 100, o custo unitário é $0,04$, caso contrário é $0,035$.
- Se o número de chamadas for maior ou igual do que 50 e menor que 500 temos que considerar o número de SMSs enviados. Se o número de SMSs for inferior a 300, então o custo unitário de uma chamada é $0,09$ e o de um SMS é $0,03$. Se o número de SMSs for maior ou igual a 300, então o custo unitário de uma chamada é $0,08$ e o de um SMS é $0,02$.
- Finalmente, se o número de chamadas for maior ou igual que 500 então o custo unitário de uma chamada é $0,06$ e os SMSs são gratuitos.

2.2 A entidade Client

Cada cliente de *Vos* tem um nome (cadeia de caracteres com um tamanho mínimo igual a 5) e um número de identificação fiscal (inteiro). Este número representa um identificador único em *Vos*. Cada cliente de *Vos* tem entre um e cinco números de telefone que têm que pertencer a *Vos*. Cada cliente pode associar um telemóvel a um dos números de telefone que lhe está atribuído. A figura 2 apresenta a interface desta entidade.

Cada cliente pode registar no sistema um determinado número de números de telefone de *amigos*. O número máximo de números de telefone de amigos que um cliente pode registar é igual ao triplo da quantidade de números de telefone pertencentes ao cliente mais 5. Os número de telefone registados como amigos têm alguns privilégios relacionados com a realização de chamadas e envio de SMSs (para mais detalhes ver secção 2.3).

2.3 A entidade Mobile

Um telemóvel pode realizar ou receber chamadas e enviar ou receber SMSs, sendo representado pela classe `Mobile`. Um telefone ligado tem dois modos (*amigo* e *silêncio*) que podem estar activos ou não. A figura 3 apresenta a interface desta entidade.

```
public class  Client {
  Client(String name, int nif, int phoneNumber) { /* .... */ }

  // Management of the phone numbers of this client
  void addPhoneNumber(int phoneNumber) throws InvalidOperationException { /* .... */ }
  void removePhoneNumber(int phoneNumber) throws InvalidOperationException  { /* .... */ }

  // registers a phoneNumber of this client to a given type of mobile phone.
  public Mobile registerMobile(int phoneNumber) throws InvalidOperationException { /* .... */ }

  // unregisters the mobile phone associated with phoneNumber
  public void unregisterMobile(int phoneNumber) throws InvalidOperationException { /* .... */ }

  // returns the mobiles of this client
  public List<Mobile> getMobiles() { /* .... */ }

  // returns the phone numbers assigned to this client.
  public List<Integer> getPhoneNumbers() { /* .... */ }

  // returns name of client
  public String getName() { /* .... */ }

  // return list of registered friends
  publi List<Integer> getFriends() { /* .... */ }

  // friends management
  public void addFriend(int phoneNumber) throws InvalidNumberPhoneException { /* .... */ }
  public void removeFriend(int phoneNumber) throws InvalidOperationException { /* .... */ }
  ...
}
```

Figura 2: A interface da classe *Client*.

De seguida descrevem-se as interacções válidas para um telemóvel. O telemóvel pode estar desligado ou desligado. Quando se cria um telemóvel, este está desligado. Um telemóvel desligado não pode receber nem fazer/enviar chamadas ou SMSs nem alterar o modo do telemóvel pelo que os únicos métodos válidos nesta situação são `turnOn` e `getStatus`. Quando se liga um telemóvel (via `turnOn`), este pode fazer chamadas, enviar SMSs e receber chamadas e SMSs e nenhum dos modos está active. É possível alterar o modo de um telemóvel ligado. Um telemóvel ligado pode estar em modo *amigo* e/ou modo *silêncio*. Se o telemóvel tiver apenas o modo *silêncio* activo, então apenas pode enviar e receber SMSs. Se tiver apenas o modo *amigo* activo, então pode fazer chamadas, enviar SMSs mas apenas pode receber chamadas e SMSs de números registados como amigos do cliente associado ao telemóvel. Finalmente, se o telemóvel estiver ambos os modos activos, então pode enviar SMSs e só pode receber SMSs de números registados como amigos do cliente associado ao telemóvel. O modo de um telemóvel ligado pode ser alterado em qualquer instante. Um telemóvel ligado pode ser desligado em qualquer momento. O método *getStatus* permite saber o estado do telemóvel (se está ligado ou não e quais os modos activos) em qualquer instante. Invocações inválidas dos métodos de `Mobile` (em situações não descritas anteriormente) não deverão ter qualquer efeito e deverão lançar a excepção *InvalidOperationException*.

3 Avaliação do projecto

Todos os casos de teste a realizar devem ser desenhados aplicando os padrões de desenho de testes mais apropriados. Os casos de testes a realizar por cada grupo correspondem a testar alguns métodos e classes das entidades descritas anteriormente:

- Casos de teste ao nível de classe da classe *Client*. Valem entre 0 e 3,5 valores.
- Casos de teste ao nível de classe da classe *Mobile*. Valem entre 0 e 6,5 valores.
- Casos de teste correspondentes ao método *assignPhonenumber* da classe *ClientManager*. Valem entre 0 e 3,5 valores.

```
public class  Mobile {
  // Creates a mobile with the specified phone number associated with the specified client.
  public Mobile(Client client, int phoneNumber) { /* .... */ }

  // Turns on the mobile with both modes disabled.
  public void turnOn() throwns InvalidOperationException { /* .... */ }

  // Turns off the mobile.
  public void turnOff() throwns InvalidOperationException { /* .... */ }

  // Changes the silent mode of a turned on mobile.
  public void setSilentMode(boolean silentMode) throwns InvalidOperationException { /* .... */ }

  // Changes the friend mode of a turned on mobile.
  public void setFriendMode(boolean friendMode) throwns InvalidOperationException { /* .... */ }

  // Sends an SMS to the specified phone number,
  void sendSMS(String smsMessage, int calledPhoneNumber) throws InvalidPhoneNumberException,
     InvalidOperationException { /* .... */ }

  // Receives an SMS from the specified phone number
  void receiveSMS(String smsMessage, int callerPhoneNumber) throwns InvalidOperationException { /* .... */ }

  // Makes a call to the specified phone number.
  void makeCall(int calledPhoneNumber) throws InvalidPhoneNumberException, InvalidOperationException
  { /* .... */ }

  // Accepts a phone call from callerPhoneNumber
  void acceptCall(int callerPhoneNumber) throwns InvalidOperationException { /* .... */ }

  // Returns a string describing the status of this mobile phone: turned off/on and silent/friend mode
  public String getStatus() { /* .... */ }

  ....
}
```

Figura 3: A interface da classe *Mobile*.

- Casos de teste correspondentes ao método *computeBill* da classe *ClientManager*. Valem entre 0 e 3,5 valores.
- Adicionalmente, é necessário concretizar 6 casos de teste da bateria de testes desenhada para exercitar a classe *Client*. Esta concretização deve ser feita utilizando a framework de testes *TestNG* e vale entre 0 e 3 valores.

Para cada método ou classe a testar é necessário indicar o seguinte:

- O nome do padrão de teste aplicado.
- Caso seja aplicável, indicar o resultado dos vários passos da aplicação do padrão, utilizando o formato apresentado nas aulas teóricas.
- A descrição dos casos de teste resultantes da aplicação do padrão de teste escolhido.

Caso algum dos pontos mencionados acima apenas seja satisfeito parcialmente a nota será dada na proporção realizada.

3.1 Discussão do projecto

Poderá existir uma discussão que avalie a capacidade dos alunos em realizar testes semelhantes aos realizados no projecto. Esta decisão cabe exclusivamente ao corpo docente da cadeira. Alunos cuja nota no primeiro teste seja inferior em mais de 5 valores à nota do projecto terão de realizar uma discussão e neste caso a nota do projecto será individualizada e dependerá do desempenho na discussão.

3.2 Detecção de cópias

A submissão de um projecto pressupõe o **compromisso de honra** que o trabalho incluso foi realizado pelos alunos referenciados nos ficheiros/documentos submetidos para avaliação. A quebra deste compromisso, ou seja a tentativa de um grupo se apropriar de trabalho realizado por colegas, tem como consequência a reprovação nesta disciplina de todos os alunos envolvidos (incluindo os que possibilitaram a ocorrência).

3.3 Notas Finais

O prazo de entrega do projecto é dia **6 de Abril de 2018 às 17:00**. O protocolo de entrega do projecto será disponibilizado oportunamente na página da cadeira bem como qual a documentação a entregar.

Toda a informação referente a este projecto estará disponível na página da cadeira na secção *Projecto*. O esclarecimento de dúvidas deve ser feito preferencialmente nas sessões de dúvidas.

Bom Trabalho!